15-819K: Logic Programming

Lecture 17

Mode Checking

Frank Pfenning

October 26, 2006

In this lecture we present *modes* to capture directionality in logic programs. Mode checking helps to catch programming errors and also allows for more efficient implementation. It is based on a form of *abstract interpretation* where substitutions are approximated by a two-element lattice: completely unknown terms, and terms known to be ground.

17.1 Modes

We have already informally used *modes* in the discussion of logic programs. For example, we can execute certain predicates in multiple directions, but this is impossible in other directions. We use modes + for *input*, − for *output* and $*$ for *bi-directional* arguments that are not designated as input or output.

We define the meaning as follows:

- Input arguments (+) must be ground when a predicate is invoked. It is an error if the mode analysis cannot establish this for a call to a predicate.

- Output arguments (−) must be ground when a predicate succeeds. It is an error if the mode analysis cannot establish this for the definition of a predicate.

From this we can deduce immediately:

- Input arguments (+) can be *assumed* to be ground when analyzing the definition of a predicate.

- Output arguments (`-`) can be *assumed* to be ground after a call to a predicate returns.

As an example, we return to our old program for addition.

```
plus(z, N, N).
plus(s(M), N, s(P)) :- plus(M, N, P).
```

First we check the mode

```
plus(+, +, -)
```

Consider the first clause, `plus(z, N, N)`. We are permitted to assume that the first two argument are ground, hence `N` is ground. We have to show the third argument, `N` is ground, which we just established. Therefore, the first clause is well-moded.

Looking at the head of the second clause, we may assume that `M` and `N` are ground. We have to eventually show the `s(P)` will be ground upon success. Now we analyze the body of the clause, `plus(M, N, P)`. This call is well-moded, because both `M` and `N` are known to be ground. Conversely, we may now assume that the output argument `P` is ground. Consequently `s(P)` is ground as required and the second clause is well-moded. Hence the definition of `plus` has the indicated mode.

I suggest you walk through showing that `plus` as has modes

```
plus(+, -, +)
plus(-, +, +)
```

which are two ways to calculate the difference of two numbers.

On the other hand, it does not have mode

```
plus(+, -, -)
```

because, for example, `plus(z, N, P)` succeeds without grounding either the second or third argument.

Modes are useful for a variety of purposes. First of all, they help to catch program errors, just like types. If a predicate does not satisfy an expected mode, it is likely a bug. This can happen fairly easily when names of free variables in clauses are mistyped. It is standard practice for Prolog compilers to produce a warning if a variable is used only once in a clause, under the assumption that it is a likely source of bugs. Unfortunately, this produces many false positives on correct programs. Mode errors are a much more reliable indication of bugs.

Secondly, modes can be used to produce more efficient code in a compiler, for example, by optimizing away occurs-checks (presuming you are interested in sound unification, as you should be), or by producing more efficient code for matching a goal against clause heads.

Thirdly, modes are helpful as a precondition for other analyses, such as termination analysis. For example, the `nat` predicate defined by

```
nat(s(N)) :- nat(N).
nat(z).
```

will terminate if the argument is ground, but diverge if the argument is a variable.

17.2 Semantic Levels

As laid out in the previous lecture, we need to consider which kind of semantics is appropriate for defining mode analysis? We would like to stay as high-level as possible, while still being able to express the property in question. Because mode analysis depends on subgoal order, one would expect to make subgoal order explicit. Moreover, since groundedness is a property related to the substitution that is applied, we also should make a substitution explicit during computation. On the other hand, modes do not interact with backtracking, so we don't expect to need a failure continuation.

We take here a slight shortcut, using a semantics with an explicit substitution, but *not* a subgoal stack. The resulting soundness property for mode analysis is not as strong as we may wish, as discussed in Exercise 17.1, but it is a bit easier to manage because it is easier to understand the concept of a successful return of a predicate call.

17.3 A Substitution Semantics

The semantics we gave in the previous lecture takes a goal G under a substitution τ. It produces a substitution θ with the invariant that $G\tau[\theta\sigma]$ for any grounding substitution σ. We define it on the fully residuated form, where for every predicate p there is exactly one clause, and this clause has the form $\forall \mathbf{x}, \mathbf{y}, \mathbf{z}.\, p(\mathbf{x}, \mathbf{y}, \mathbf{z}) \leftarrow G$ where $\mathbf{x}$ are the input arguments, $\mathbf{y}$ are the bi-directional arguments, and $\mathbf{z}$ are the output arguments of p. We have collected them into left-to-right form only for convenience, although this is probably also good programming practice. The rules are summarized in Figure 1.

$$\frac{\tau \vdash G_1 \;/\; \theta_1 \quad \tau[\theta_1] \vdash G_2 \;/\; \theta_2}{\tau \vdash G_1 \wedge G_2 \;/\; \theta_1\theta_2} \qquad \frac{}{\tau \vdash \top \;/\; (\cdot)}$$

$$\frac{(\forall \mathbf{x}, \mathbf{y}, \mathbf{z}.\, p(\mathbf{x}, \mathbf{y}, \mathbf{z}) \leftarrow G) \in \Gamma \quad \mathbf{t}\tau/\mathbf{x}, \mathbf{r}\tau/\mathbf{y}, \mathbf{s}\tau/\mathbf{z} \vdash G \;/\; \theta}{\tau \vdash p(\mathbf{t}, \mathbf{r}, \mathbf{s}) \;/\; \theta}$$

$$\frac{\tau \vdash G_1 \;/\; \theta}{\tau \vdash G_1 \vee G_2 \;/\; \theta} \qquad \frac{\tau \vdash G_2 \;/\; \theta}{\tau \vdash G_1 \vee G_2 \;/\; \theta} \qquad \begin{array}{c}\text{no rule for}\\ \tau \vdash \bot \;/\; \_\end{array}$$

$$\frac{X \notin \mathrm{FV}(\tau) \quad \tau, X/x \vdash G \;/\; \theta}{\tau \vdash \exists x.\, G \;/\; \theta} \qquad \frac{\mathbf{t}\tau \doteq \mathbf{s}\tau \mid \theta}{\tau \vdash \mathbf{t} \doteq \mathbf{s} \;/\; \theta}$$

Figure 1: Substitution Semantics

17.4 Abstract Substitutions

One common way of designing a program analysis is to construct and abstraction of a concrete domain involved in the operational semantics. Here, we abstract away from the substitution terms, tracking only if the terms are ground g or unknown u. They are related by an information ordering in the sense that u has no information and g is the most information we can have about a term. We write this in the form of a partial order which, in this case, is rather trivial.

$$\begin{tikzcd} \mathsf{u} \arrow[d, dash] \\ \mathsf{g} \end{tikzcd}$$

We write $\mathsf{g} \leq \mathsf{u}$. If we are lower in this partial order we have more information; higher up we have less. This order is consistent with the interpretation of g and u as sets of terms: g is the set of ground terms, which is a subset of the set of all terms u.

Other forms of abstract interpretation, or a more detailed mode analysis, demands more complex orderings. We will see in the analysis which form of operations are required to be defined on the structure.

Abstract substitutions now have the form

$$\hat{\tau} ::= \cdot \mid \hat{\tau}, \mathsf{u}/x \mid \hat{\tau}, \mathsf{g}/x.$$

An abstract substitution $\hat{\tau}$ *approximates* an actual substitution τ if they have the same domain, and if whenever $\mathsf{g}/x \in \hat{\tau}$ it is indeed the case that $t/x \in \tau$ and $\mathrm{FV}(t) = \emptyset$. For variables marked as unknown in $\hat{\tau}$ there is no requirement placed on τ. We write $\hat{\tau} \preccurlyeq \tau$ if $\hat{\tau}$ approximates τ.

17.5 Mode Analysis Judgment

Now we abstract from the concrete operational semantics to one where we just carry abstract substitutions. The parallel nature of the operational and analysis rules leads to a mangeable proof of soundness of the analysis. Of course, completeness can not hold: there will always be programs that will respect modes at run-time, but fail the decidable judgment of well-modedness defined below (see Exercise 17.2).

At the top level we check each predicate separately. However, we assume that modes for all predicates are declared simultaneously, or at least that the modes of a predicate are defined before they are used in another predicate. The main judgment is

$$\hat{\tau} \vdash G \;/\; \hat{\sigma}$$

where $\mathrm{dom}(\tau) \supseteq \mathrm{FV}(G)$ and $\hat{\tau} \geq \hat{\sigma}$. The latter is interpreted pointwise, according to the partial order among the abstract elements. So $\hat{\tau}$ and $\hat{\sigma}$ have the same domain, and $\hat{\sigma}$ preserves all g/x in $\hat{\tau}$, but may transform some u/x into g/x.

When analyzing the definition of a predicate we are allowed to assume that all input arguments are ground and we have to show that upon success, the output arguments are ground. We do not assume or check anything about the bi-directional arguments.

$$\frac{(\forall \mathbf{x}, \mathbf{y}, \mathbf{z}.\, p(\mathbf{x}, \mathbf{y}, \mathbf{z}) \leftarrow G) \in \Gamma \qquad \mathbf{g}/\mathbf{x}, \mathbf{u}/\mathbf{y}, \mathbf{u}/\mathbf{z} \vdash G(\mathbf{x}, \mathbf{y}, \mathbf{z}) \;/\; (\mathbf{g}/\mathbf{x}, \_/\mathbf{y}, \mathbf{g}/\mathbf{z})}{p \; \mathit{wellmoded}}$$

The judgment $\hat{\tau} \vdash \mathbf{t}$ *ground* checks that all terms in $\mathbf{t}$ are ground assuming the information given in $\hat{\tau}$.

$$\frac{\mathsf{g}/x \in \hat{\tau}}{\hat{\tau} \vdash x \; \mathit{ground}} \qquad \frac{\hat{\tau} \vdash \mathbf{t} \; \mathit{ground}}{\hat{\tau} \vdash f(\mathbf{t}) \; \mathit{ground}}$$

$$\frac{}{\hat{\tau} \vdash (\cdot) \; \mathit{ground}} \qquad \frac{\hat{\tau} \vdash t \; \mathit{ground} \quad \hat{\tau} \vdash \mathbf{t} \; \mathit{ground}}{\hat{\tau} \vdash (t, \mathbf{t}) \; \mathit{ground}}$$

Besides the abstraction, the gap to bridge between the operational and abstract semantics is only that fact in $\tau \vdash G \;/\; \theta$ the output θ is an increment: we apply it to τ to obtain the substitution under which G is true. $\hat{\sigma}$ on the other hand is a completed approximate substitution. This is reflected in the following property we show in the end.

If $\tau \vdash G \;/\; \theta$ and $\hat{\tau} \preccurlyeq \tau$ and $\hat{\tau} \vdash G \;/\; \hat{\sigma}$ then $\hat{\sigma} \preccurlyeq \tau[\theta]$.

Atoms. An atomic goal represents a procedure call. We therefore have to show that the input arguments are ground and we are permitted to assume that the output arguments will subsequently be ground.

$$\frac{\hat{\tau} \vdash \mathbf{t} \; \mathit{ground} \quad \hat{\tau} \vdash \mathbf{s} \;/\; \hat{\sigma}}{\hat{\tau} \vdash p(\mathbf{t}, \mathbf{r}, \mathbf{s}) \;/\; \hat{\sigma}}$$

The judgment $\hat{\tau} \vdash \mathrm{s} \;/\; \hat{\sigma}$ refines the information in $\hat{\tau}$ by noting that all variables in s can also be assumed ground.

$$\frac{}{(\hat{\tau}_1, \_/x, \hat{\tau}_2) \vdash x \;/\; (\hat{\tau}_1, \mathrm{g}/x, \hat{\tau}_2)} \qquad \frac{\hat{\tau} \vdash \mathbf{s} \;/\; \hat{\sigma}}{\hat{\tau} \vdash f(\mathbf{s}) \;/\; \hat{\sigma}}$$

$$\frac{}{\hat{\tau} \vdash (\cdot) \;/\; \hat{\tau}} \qquad \frac{\hat{\tau}_1 \vdash s \;/\; \hat{\tau}_2 \quad \hat{\tau}_2 \vdash \mathbf{s} \;/\; \hat{\tau}_3}{\hat{\tau}_1 \vdash (s, \mathbf{s}) \;/\; \tau_3}$$

Conjunction. Conjunctions are executed from left-to-right, so we propagate the mode information in the same order.

$$\frac{\hat{\tau}_1 \vdash G_1 \;/\; \hat{\tau}_2 \quad \hat{\tau}_2 \vdash G_2 \;/\; \hat{\tau}_3}{\hat{\tau}_1 \vdash G_1 \wedge G_2 \;/\; \hat{\tau}_3}$$

Truth. Truth does not affect any variables, so the modes are simply propagated unchanged.

$$\frac{}{\hat{\tau} \vdash \top \;/\; \hat{\tau}}$$

Disjunction. Disjunction represents an interesting challenge. For a goal $G_1 \vee G_2$ we do not know which subgoal will succeed. Therefore a variable is only definitely known to be ground after execution of the disjunction, if it is known to be ground in both cases.

$$\frac{\hat{\tau} \vdash G_1 \sqcup \hat{\sigma}_1 \quad \hat{\tau} \vdash G_2 \sqcup \hat{\sigma}_2}{\hat{\tau} \vdash G_1 \vee G_2 \;/\; \hat{\sigma}_1 \sqcup \hat{\sigma}_2}$$

The least upper bound operation $\sqcup$ is applied to two abstract substitutions point-wise, and on abstract terms it is defined by

$$\begin{array}{rcl} \mathsf{g} \sqcup \mathsf{g} & = & \mathsf{g} \\ \mathsf{g} \sqcup \mathsf{u} & = & \mathsf{u} \\ \mathsf{u} \sqcup \mathsf{g} & = & \mathsf{u} \\ \mathsf{u} \sqcup \mathsf{u} & = & \mathsf{u} \end{array}$$

This is a common pattern in logic program analysis, where the least upper bound operations comes from the order on the abstract elements. To make sure that such a least upper bound always exist we generally stipulate that the order of abstract elements actually constitutes a *lattice* so that least upper bounds ($\sqcup$) and greatest lower bounds ($\sqcap$) of finite sets always exist. As a special case, the least upper bound of the empty set is the bottom element of the lattice, usually denoted by $\perp$ or 0. We use the latter because $\perp$ is already employed with its logical meaning. Here, the bottom element is g for an individual element, and $\mathbf{g}/\mathbf{x}$ for a substitution.

Falsehood. In the operational semantics there is no rule for proving falsehood. In the mode analysis, however, we need a rule for handling falsehood, since analysis should not fail unless there is a mode error. Recall that we need for the output of mode analysis to approximate the output substitution $\tau[\theta]$ if $\tau \vdash G \;/\; \theta$. But $G = \perp$ can never succeed, so this requirement is vacuous. Consequently, is it safe to pick the bottom element of the lattice.

$$\frac{\mathrm{dom}(\hat{\tau}) = \mathbf{x}}{\hat{\tau} \vdash \perp \;/\; (\mathbf{g}/\mathbf{x})}$$

Existential. Solving an existential creates a new logic variable. This is still a free variable, so we mark its value as not known to be ground.

$$\frac{(\hat{\tau}, \mathsf{u}/x) \vdash G \;/\; (\hat{\sigma}, \_/x)}{\hat{\tau} \vdash \exists x.\, G \;/\; \hat{\sigma}}$$

Since there is no requirement that existential variables eventually become ground, we do not care what is known about the substitution term for x upon the successful completion of G.

Equality. When encountering an equality we need to descend into the terms, abstractly replaying unification, to approximate the resulting substitution. We therefore distinguish various cases. Keep in mind that analysis should always succeed, even if unification fails at runtime, which gives us more cases to deal with than one would initially expect. We write $\hat{\tau} + \mathsf{g}/x$ for the result of setting the definition for x in $\hat{\tau}$ to g.

$$\frac{\hat{\tau} \vdash s \; \mathit{ground}}{\hat{\tau} \vdash x \doteq s \;/\; \hat{\tau} + \mathsf{g}/x} \qquad \frac{\begin{array}{c}\mathsf{g}/x \in \hat{\tau} \\ \hat{\tau} \nvdash s \; \mathit{ground} \quad \hat{\tau} \vdash s \;/\; \hat{\sigma}\end{array}}{\hat{\tau} \vdash x \doteq s \;/\; \hat{\sigma}} \qquad \frac{\begin{array}{c}\mathsf{u}/x \in \hat{\tau} \\ \hat{\tau} \nvdash s \; \mathit{ground}\end{array}}{\hat{\tau} \vdash x \doteq s \;/\; \hat{\tau}}$$

$$\frac{\hat{\tau} \vdash t \; \mathit{ground}}{\hat{\tau} \vdash t \doteq y \;/\; \hat{\tau} + \mathsf{g}/y} \qquad \frac{\begin{array}{c}\mathsf{g}/y \in \hat{\tau} \\ \hat{\tau} \nvdash t \; \mathit{ground} \quad \hat{\tau} \vdash t \;/\; \hat{\sigma}\end{array}}{\hat{\tau} \vdash t \doteq y \;/\; \hat{\sigma}} \qquad \frac{\begin{array}{c}\mathsf{u}/y \in \hat{\tau} \\ \hat{\tau} \nvdash t \; \mathit{ground}\end{array}}{\hat{\tau} \vdash t \doteq y \;/\; \hat{\tau}}$$

$$\frac{\hat{\tau} \vdash \mathbf{t} \doteq \mathbf{s} \;/\; \hat{\sigma}}{\hat{\tau} \vdash f(\mathbf{t}) \doteq f(\mathbf{s}) \;/\; \hat{\sigma}} \qquad \frac{f \neq g \quad \mathrm{dom}(\hat{\tau}) = \mathbf{x}}{\hat{\tau} \vdash f(\mathbf{t}) \doteq g(\mathbf{s}) \;/\; (\mathbf{g}/\mathbf{x})}$$

$$\frac{\hat{\tau}_1 \vdash t \doteq s \;/\; \hat{\tau}_2 \quad \hat{\tau}_2 \vdash \mathbf{t} \doteq \mathbf{s} \;/\; \hat{\tau}_3}{\hat{\tau}_1 \vdash (t, \mathbf{t}) \doteq (s, \mathbf{s}) \;/\; \hat{\tau}_3} \qquad \frac{}{\hat{\tau} \vdash (\cdot) \doteq (\cdot) \;/\; \hat{\tau}}$$

$$\frac{\mathrm{dom}(\hat{\tau}) = \mathbf{x}}{\hat{\tau} \vdash (\cdot) \doteq (s, \mathbf{s}) \;/\; (\mathbf{g}/\mathbf{x})} \qquad \frac{\mathrm{dom}(\hat{\tau}) = \mathbf{x}}{\hat{\tau} \vdash (t, \mathbf{t}) \doteq (\cdot) \;/\; (\mathbf{g}/\mathbf{x})}$$

The first and second lines overlap in the sense that for some equations, more than one rule applies. However, the answer is the same in either case. We could resolve the ambiguity by requiring, for example, that in the second line t is not a variable, that is, of the form $f(\mathbf{t})$.

17.6 Soundness

Next we have to prove soundness of the analysis. First we need a couple of lemmas regarding the term-level judgments. One can view these as encoding what happens when an approximate substitution is applied, so we refer to them as the first and second approximate substitution lemma. To see how they arise you might analyze the soundness proof below first.

Lemma 17.1 *If $\hat{\tau} \vdash t$ ground and $\hat{\tau} \preccurlyeq \tau$ and then $t\tau$ ground.*

Proof: By induction on the structure of $\mathcal{D}$ of $\hat{\tau} \vdash t$ *ground*. □

Lemma 17.2 *If* $\hat{\tau} \vdash s \ / \ \hat{\sigma}$ *and* $\hat{\tau} \preccurlyeq \tau$ *and* $s\tau[\theta]$ *ground then* $\hat{\sigma} \preccurlyeq \tau[\theta]$.

Proof: By induction on the structure of $\mathcal{D}$ of $\hat{\tau} \vdash s \ / \ \hat{\sigma}$. □

Theorem 17.3 *If* $\tau \vdash G \ / \ \theta$ *and* $\hat{\tau} \preccurlyeq \tau$ *and* $\hat{\tau} \vdash G \ / \ \hat{\sigma}$ *then* $\hat{\sigma} \preccurlyeq \tau[\theta]$.

Proof: By induction on the structure of $\mathcal{D}$ of $\tau \vdash G \ / \ \theta$, applying inversion to the given mode derivation in each case.

Case: $\mathcal{D} = \dfrac{(\forall \mathbf{x}, \mathbf{y}, \mathbf{z}.\, p(\mathbf{x}, \mathbf{y}, \mathbf{z}) \leftarrow G) \in \Gamma \quad (\mathbf{t}^+\tau/\mathbf{x}, \mathbf{r}^*\tau/\mathbf{y}, \mathbf{s}^-\tau/\mathbf{z}) \vdash G \ / \ \theta}{\tau \vdash p(\mathbf{t}^+, \mathbf{r}^*, \mathbf{s}^-) \ / \ \theta}$.

$\hat{\tau} \vdash p(\mathbf{t}^+, \mathbf{r}^*, \mathbf{s}^-) \ / \ \hat{\sigma}$	Assumption
$\hat{\tau} \vdash \mathbf{t}^+$ *ground* and	
$\hat{\tau} \vdash \mathbf{s}^- \ / \ \hat{\sigma}$	By inversion
$\hat{\tau} \preccurlyeq \tau$	Assumption
$\mathbf{t}^+\tau$ *ground*	By approx. subst. lemma
$(\mathbf{g}/\mathbf{x}, \mathbf{u}/\mathbf{y}, \mathbf{u}/\mathbf{z}) \preccurlyeq (\mathbf{t}^+\tau/\mathbf{x}, \mathbf{r}^*\tau/\mathbf{y}, \mathbf{s}^-\tau/\mathbf{z})$	From previous line
p *wellmoded*	Assumption
$\mathbf{g}/\mathbf{x}, \mathbf{u}/\mathbf{y}, \mathbf{u}/\mathbf{z} \vdash G \ / \ (\mathbf{g}/\mathbf{x}, \_/\mathbf{y}, \mathbf{g}/\mathbf{z})$	By inversion
$(\mathbf{g}/\mathbf{x}, \_/\mathbf{y}, \mathbf{g}/\mathbf{z}) \preccurlyeq (\mathbf{t}^+\tau\theta/\mathbf{x}, \mathbf{r}^*\tau\theta/\mathbf{y}, \mathbf{s}^-\tau\theta/\mathbf{z})$	By ind.hyp.
$\mathbf{s}^-\tau\theta$ *ground*	By defn. of $\preccurlyeq$
$\hat{\sigma} \preccurlyeq \tau\theta$	By approx. subst. lemma

Case: $\mathcal{D} = \dfrac{\tau \vdash G_1 \ / \ \theta_1 \quad \tau\theta_1 \vdash G_2 \ / \ \theta_2}{\tau \vdash G_1 \wedge G_2 \ / \ \theta_1\theta_2}$.

$\hat{\tau} \vdash G_1 \wedge G_2 \ / \ \hat{\sigma}_2$	Assumption
$\hat{\tau} \vdash G_1 \ / \ \hat{\sigma}_1$ and	
$\hat{\sigma}_1 \vdash G_2 \ / \ \hat{\sigma}_2$ for some $\hat{\sigma}_1$	By inversion
$\hat{\tau} \preccurlyeq \tau$	Assumption
$\hat{\sigma}_1 \preccurlyeq \tau[\theta_1]$	By ind.hyp.
$\hat{\sigma}_2 \preccurlyeq (\tau[\theta_1])[\theta_2]$	By ind.hyp.
$\hat{\sigma}_2 \preccurlyeq \tau[\theta_1\theta_2]$	By assoc of composition

Case: $\mathcal{D} = \dfrac{}{\tau \vdash \top \ / \ (\cdot)}$.

$\hat{\tau} \vdash \top \ / \ \hat{\sigma}$	Assumption
$\hat{\sigma} = \hat{\tau}$	By inversion
$\hat{\tau} \preccurlyeq \tau$	Assumption
$\hat{\sigma} \preccurlyeq \tau[\cdot]$	By identity of $(\cdot)$

Case: $\mathcal{D} = \dfrac{\tau \vdash G_1 \mathrel{/} \theta}{\tau \vdash G_1 \vee G_2 \mathrel{/} \theta}$.

$\hat{\tau} \vdash G_1 \vee G_2 \mathrel{/} \hat{\sigma}$	Assumption
$\hat{\tau} \vdash G_1 \mathrel{/} \hat{\sigma}_1$ and	
$\hat{\tau} \vdash G_2 \mathrel{/} \hat{\sigma}_2$ for some $\hat{\sigma}_1$, $\hat{\sigma}_2$ with $\hat{\sigma} = \hat{\sigma}_1 \sqcup \hat{\sigma}_2$	By inversion
$\hat{\tau} \preccurlyeq \tau$	Assumption
$\hat{\sigma}_1 \preccurlyeq \tau[\theta]$	By ind.hyp.
$\hat{\sigma}_1 \sqcup \hat{\sigma}_2 \preccurlyeq \tau[\theta]$	By property of least upper bound

Case: $\mathcal{D} = \dfrac{\tau \vdash G_2 \mathrel{/} \theta}{\tau \vdash G_1 \vee G_2 \mathrel{/} \theta}$. Symmetric to the previous case.

Case: $\tau \vdash \bot \mathrel{/} \theta$. There is no rule to conclude such a judgment. Therefore the property holds vacuously.

Case: $\mathcal{D} = \dfrac{\tau, X/x \vdash G \mathrel{/} \theta \quad X \notin \mathrm{FV}(\tau)}{\tau \vdash \exists x.\, G \mathrel{/} \theta}$.

$\hat{\tau} \vdash \exists x.\, G \mathrel{/} \hat{\sigma}$	Assumption
$\hat{\tau}, \mathsf{u}/x \vdash G \mathrel{/} (\hat{\sigma}, \_/x)$	By inversion
$\hat{\tau} \preccurlyeq \tau$	Assumption
$(\hat{\tau}, \mathsf{u}/x) \preccurlyeq (\tau, X/x)$	By defn. of $\preccurlyeq$
$(\hat{\sigma}, \_/x) \preccurlyeq (\tau, X/x)[\theta]$	By ind.hyp.
$\hat{\sigma} \preccurlyeq \tau[\theta]$	By prop. of subst. and approx.

Case: $\mathcal{D} = \dfrac{\mathbf{t}\tau \doteq \mathbf{s}\tau \mid \theta}{\tau \vdash \mathbf{t} \doteq \mathbf{s} \mathrel{/} \theta}$. This case is left to the reader (see Exercise 17.3).

□

17.7 Exercises

Exercise 17.1 *One problem with well-modedness in this lecture is that we only prove that if a well-moded query succeeds then the output will be ground. A stronger property would be that during the execution of the program, every goal and subgoal we consider will be well-moded. However, this requires a transition semantics and a different soundness proof.*

Write a suitable operational semantics and prove soundness of mode checking in the sense sketched above. This is a kind of mode preservation theorem, analogous to a type preservation theorem.

Exercise 17.2 *Give a program that respects modes at run-time in the sense that*

- *input arguments (+) are always ground when a predicate is invoked, and*
- *output arguments (−) are always ground when a predicate succeeds,*

and yet is not well-moded according to our analysis.

Exercise 17.3 *Complete the proof of soundness of mode analysis by giving the case for unification. If you need a new lemma in addition to the approximate substitution lemmas, carefully formulate and prove them.*